Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Оренбургский государственный университет»

Кафедра промышленной электроники и
информационно-измерительной техники

А.В. Хлуденев

ОТЛАДОЧНЫЕ СРЕДСТВА МИКРОПРОЦЕССОРНЫХ СИСТЕМ

Методические указания

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательной программе высшего образования по направлению подготовки 11.03.04 Электроника и наноэлектроника

Оренбург
2018

УДК 004.4 : 681.3.068
ББК 32.973.26
Х60

Рецензент - кандидат технических наук, доцент С.А. Сильвашко

Хлуденев, А.В.
Х60 Отладочные средства микропроцессорных систем: методические указания/ А.В. Хлуденев; Оренбургский гос. ун-т. – Оренбург: ОГУ, 2018.

Методические указания содержат рекомендации по выполнению лабораторных работ по дисциплине «Отладочные средства микропроцессорных систем».

Методические указания предназначены для обучающихся по образовательной программе высшего образования по направлению подготовки 11.03.04 Электроника и наноэлектроника.

ББК 32.973.26
УДК 004.4 : 681.3.068

Содержание

1 Отладка простых программ

1.1 Описание работы

Цель работы**:**

- освоить работу в среде системы MPLAB IDE;

- приобрести практические навыки отладки простых программ.

Объектом отладки в работе является программа генератора чисел. Числовая последовательность (в соответствии с вариантом задания) формируется на линиях порта PORTD.

MPLAB IDE – это интегрированная среда разработки программ для микроконтроллеров (МК) PICmicro фирмы Microchip, позволяющая писать, компилировать и отлаживать программы [1]. MPLAB IDE включает несколько инструментальных средств:

- *Project Manager* (менеджер проекта) необходим, чтобы создать проект и работать с файлами, связанными с проектом;

- *Editor* (редактор) необходим для создания и редактирования исходных текстов программ, а также используется как окно отладчика;

- *Language Tools* (языковый инструмент) – позволяет работать с кросс- ассемблером MPASM или кросс-компиляторами;

- *Debugger* (отладчик) – совместно с редактором реализует механизмы управляемости и наблюдаемости процесса отладки целевых программ;

- *Execution Engines* (механизм исполнения команд) – программные симуляторы, позволяющие выполнять команды МК на персональном компьютере и имитировать работу периферийных модулей.

Подобно другим инструментальным средствам разработки программ MPLAB IDE содержит встроенный отладчик программ. Но в отличие от резидентных систем в процессе отладки используется не процессор компьютера, а МК PICmicro или его программно-логическая модель. В системе MPLAB IDE реализованы следующие режимы работы отладчика:

- *MPLAB-SIM Simulator* – отладка средствами симулятора, который моделирует работу процессорного ядра и периферийных модулей МК PICmicro;

- *MPLAB-ICE Emulator* – отладка средствами внутрисхемного эмулятора (аппаратного блока для замещения МК PICmicro в реальном времени в отлаживаемой системе);

- *MPLAB-ICD Debugger* - отладка средствами внутрисхемного отладчика (аппаратного блока, позволяющего использовать встроенные в МК PICmicro средства отладки в реальном времени в отлаживаемой системе).

Основное окно системы MPLAB IDE выглядит аналогично большинству инструментальных приложений Windows. Вверху расположена панель меню, а внизу строка состояния. В строке состояния отображается текущая настройка и состояние отладочной системы. Для разработки доступны все инструменты, в том числе для отладки исполняемого кода.

1.2 Создание проекта и трансляция программы

В процессе разработки программы формируется проект - множество взаимосвязанных файлов, включая исходный текст программы <имя>.asm, загрузочный модуль <имя>.hex и другие файлы. Все эти файлы удобно хранить в отдельной папке. Взаимосвязи файлов содержатся в корневом файле проекта.

В папке PIC создайте свою рабочую папку, а в ней папку lr1. Скопируйте в нее подготовленный файл с исходным текстом программы lr1.asm. Создайте новый проект, для этого выберите команду *Project > Project Wizard* и последовательно выполните действия:

- задайте тип микроконтроллера PIC16F877A;

- задайте транслятор Microchip MPASM Toolsuite;

- задайте имя проекта lr1 и папку для нового проекта;

- включите в проект исходный текст программы lr1.asm.

Откройте окно с исходным текстом программы, для этого в окне менеджера проекта lr1.mcw дважды щелкните по ярлыку lr1.asm в папке *Source Files* или выполните команду *File>Open*.

Выполните трансляцию программы (команда *Project>Make*). На закладке *Build* в окне *Output* будет приведена командная строка, посланная ассемблеру, сообщения об ошибках или успешном завершении трансляции.

Если программа содержала синтаксические и другие формальные ошибки, в окне будет выведен список сообщений об ошибках (ERROR). Если дважды щелкнуть мышкой на сообщении об ошибке, то курсор автоматически перейдет к соответствующему оператору в исходном тексте. Если в программе нет подобных ошибок, ассемблер выдаст сообщение BUILD SUCCEEDED. Это означает, что получен загрузочный модуль программы, готовый для выполнения микроконтроллером. Но это не означает, что программа не содержит логических ошибок, и будет работать правильно. Проверку работы программы, поиск и исправление возможных ошибок выполняют в ходе отладки.

1.3 Отладка программы

В данной лабораторной работе предусмотрено выполнение только автономной отладки программы на программно-логической модели МК. Активировать инструмент *MPLAB-SIM Simulator* можно командой *Debugger>Select Tool>MPLAB SIM*.

Существуют различные методы тестирования программ, направленные на проверку их правильности и выявление возможных ошибок. Но практически все они основаны на отслеживании выполнения программы и сравнении с ожидаемым ходом выполнения. Для отладки программ обычно применяют два режима:

- пошаговая отладка;
- выполнение программы (прогон) до точки останова.

При пошаговой отладке выполняется один оператор программы, и затем контролируются те переменные, на которые должен воздействовать данный оператор.

Отдельный участок программы можно выполнить в режиме прогона. Точку останова устанавливают на операторе, при выполнении которого необходимо проверить значения переменных или просто проконтролировать, передаётся ли управление данному оператору.

Для отслеживания содержимого ограниченного набора ячеек памяти удобно использовать смотровое окно. Чтобы создать смотровое окно выполните команду *View> Watch*. В верхней части окна *Watch* находятся прокручиваемые списки регистров специальных функций (SFR) и идентификаторов, используемых в программе (Symbol). Из левого списка выберите ячейку PORTD, добавьте в окно наблюдения нажатием кнопки *Add SFR*. Из правого списка выберите ячейку Cnt, добавьте в окно наблюдения нажатием кнопки *Add Symbol*.

Инициализируйте МК нажатием кнопки *Reset*. Программный счетчик будет сброшен в ноль. Средствами символьного отладчика строка исходного кода, соответствующая этому адресу, отмечена указателем (зеленой стрелкой). В строке состояния MPLAB IDE отображаются:

- режим работы отладчика (MPLAB SIM);
- тип МК (PIC16F877A);
- содержимое программного счетчика PC;
- содержимое аккумулятора W;
- состояние флагов Z dc c (заглавной буквой – установлен, строчной – сброшен);
- тактовая частота МК (12 MHz);
- активный банк памяти данных (bank 0).

Нажмите кнопку *Step Into*. Будет выполнена первая команда программы. Программный счетчик будет показывать PC:0x01, указатель переместится на следующую исполняемую команду. Для пошагового выполнения можно также использовать функциональную клавишу <F7>. Нажимайте клавишу <F7> несколько раз и наблюдайте, как перемещается по тексту указатель, изменяются показания программного счетчика и значения переменных в смотровом окне.

Для регистрации значений переменных в процессе прогона программы можно использовать память трассы. Установите курсор на строке с оператором goto Loop. Нажмите правую кнопку мыши и в появившемся меню выберите опцию *Set Break-point* (точка останова). Слева от этой строки появится символ “B” красного цвета, сигнализируя, что сразу перед исполнением команды в этой строке выполнение программы остановится.

Продолжите выполнение программы в режиме прогона командой *Run* или клавишей <F9>. При достижении точки останова симулятор остановится на выделенном операторе (но эта команда не выполнится).

Открыть окно памяти трассы можно командой *View>Simulator Trace* . Каждая запись памяти трассы содержит:

- адрес команды (Addr);
- код команды (Op);
- метку (Label);
- дизассемблерное представление команды (Instruction);
- адрес и содержимое ячейки источника (SA и SD);
- адрес и содержимое ячейки приемника (DA и DD);
- текущее время в циклах (Cycles).

Содержимое буфера трассировки можно сохранить в файле командой *Output to File*, очистить буфер трассировки можно командой *Clear Trace Buffer* (панель команд открывается правой кнопкой мыши).

Если нажать клавишу <F9> еще раз, симуляция программы продолжится до следующего достижения точки останова. Разумеется, когда программа выполняется и не достигла точки останова, ее можно остановить командой *Halt* или нажатием клавиши <F5>.

По содержимому окна *Watch* и буфера трассировки определите корректность выполнения программы. При выявлении несоответствия хода выполнения программы ожидаемому, необходимо локализовать ошибку в программе. Для этого необходимо:

- определить шаг (и соответствующий оператор программы), начиная с которого выполнение программы отклоняется от ожидаемого;

- проанализировать причину этого отклонения;

- исправить ошибку в исходном тексте;

- выполнить трансляцию программы.

При последующем тестировании необходимо убедиться в корректности выполнения программы. После выполнения работы закройте проект командой *Project>Close*.

1.4 Содержание отчета

Отчет должен содержать:

- схему алгоритма программы;

- исходный текст программы;

- листинги трансляции, карту памяти;

- трассу программы;

- выводы о корректности программы.

1.5 Контрольные вопросы

Перечислите основные инструментальные средства и режимы работы отладчика системы MPLAB IDE.

Перечислите основные режимы отладки программ в среде MPLAB IDE?

Поясните по схеме алгоритма работу программы.

Поясните содержание трассы программы?

Перечислите условия контролепригодности.

Приведите примеры использованных средств, обеспечивающих выполнение условий контролепригодности.

2 Отладка программы дискретного ввода-вывода

2.1 Описание работы

Цель работы - получить практические навыки выполнения автономной и комплексной отладки программ дискретного ввода-вывода в среде MPLAB IDE.

Содержанием работы является отладка программы для МК типа PIC16F877A, которая анализирует состояние кнопки, подключенной к линиям порта RC1. Нажатие с последующим отпусканием кнопки RC1 вызывает модификацию содержимого ячейки Cnt (в соответствии с вариантом задания), которое отображается на светоизлучающих диодах, подключенных к линиям порта PORTD. Активному (нажатому) состоянию кнопки соответствует низкий потенциал на линии порта PORTC. Пассивному (отжатому) состоянию кнопки соответствует высокий потенциал, который обеспечивается подтягивающими резисторами.

Для отладки программы предлагается использовать технологию нисходящей отладки. Сначала выполняется отладка основной программы, а затем подпрограмм.

2.2 Создание проекта и трансляция программы

2.2.1 В папке PIC создайте свою рабочую папку. Скопируйте в нее подготовленный файл с исходным текстом программы lr2.asm. Создайте новый проект с помощью мастера *Project>Project Wizard*. Задайте имя проекта lr2, включите в проект исходный текст программы lr2.asm.

2.2.2 Откройте окно с исходным текстом программы. При выполнении автономной отладки основной программы необходимо закомментировать оператор вызова подпрограммы задержки D5ms. Выполните трансляцию программы (команда *Project>Make*). Если программа содержала синтаксические и другие формальные ошибки, исправьте их и повторите трансляцию.

2.3 Отладка программы

2.3.1 Активируйте отладчик MPLAB IDE для работы с симулятором (команда *Debugger>Select Tool>MPLAB SIM*). Настройте симулятор для выполнения автономной отладки данного приложения (команда *Debugger>Setting*). В окне *Simulator Setting* на закладке *Osc/Trace* задайте значение тактовой частоты *Processor Frequency* 12 MHz.

Выполните команду *View> Watch*, чтобы создать окно наблюдения интересующих ячеек регистрового файла: PORTC, PORTD, Cnt.

2.3.2 Для автономной отладки программы необходимо иметь средства имитации нажатия кнопок. Для этих целей можно использовать асинхронные стимулы. Чтобы создать асинхронный стимул, выберите пункт меню *Debugger> Stimulus>New Workbook* для открытия диалогового окна настройки стимулов. В открывшемся окне *Stimulus* на закладке *Asynch* содержится таблица, каждая строка которой позволяет задать атрибуты одного стимула. Установите курсор на первой строке в столбце *Pin/SFR* и задайте линию порта ввода RC1. В столбце *Action* выберите значение *Set High* (высокий уровень). Это значение будет имитировать действие подтягивающего резистора. В следующей строке таблицы создайте стимул для RC1 для имитации воздействия кнопки. Для этого в столбце *Action* выберите значение *Pulse Low* (импульс низкого потенциала), а в столбце *Width* значение длительности импульса в единицах времени, указанных в столбце *Units*. Длительность этих импульсов в реальном устройстве определяется временем нажатия кнопки и составляет доли секунды. При моделировании целесообразно задать существенно меньшие значения, чтобы сократить время моделирования. Вполне достаточно установить 100 сус.

2.3.3 Необходимо выполнить тестирование программы. Сначала следует задать исходное (пассивное) значение разряда RC1=1. Произвольно изменить содержимое PORTC не удастся, поэтому произведите сброс МК (*Debugger> Reset*), затем нажмите кнопку *Fire* стимула RC1 типа *Set High*, выполните одну команду программы (*Debugger>Step Into*). Проконтролируйте установку значения разряда 1 порта PORTC в окне наблюдения.

Проверку работы программы удобно выполнять в режиме анимации (быстрого пошагового режима), который активируется командой *Debugger >Animate*. В этом режиме после выполнения очередной команды обновляется содержимое всех открытых окон системы MPLAB IDE и только затем управление передается следующей команде программы.

Нажмите кнопку *Fire* стимула RC1 типа *Pulse Low* на панели контроллера стимулов. Проследите за изменением соответствующего разряда PORTC в окне наблюдения. Убедитесь, что «нажатие» кнопки вызывает необходимую модификацию переменной Cnt. Повторите эту проверку необходимое число раз. Занесите в отчет формируемую последовательность значений переменной Cnt. Остановите процесс выполнения программы нажатием кнопки *Halt*.

2.3.4 Далее по плану выполняется отладка подпрограмм INT и D5ms. Восстановите операторы вызова подпрограммы задержки D5ms в исходном тексте программы. Выполните трансляцию программы.

Контролировать сохранение точки возврата можно в окне стека. Откройте окно стека командой *View>Hardware Stack*. Для контроля формируемых интервалов времени откройте окно секундомера *Debugger>Stopwatch*.

2.3.5 Чтобы проконтролировать возникновение и обработку прерываний в режиме анимации, потребуется слишком много времени. Более рационально использовать режим прогона до точки останова. Установите точку останова на векторе прерывания. Запустите выполнение программы командой *Debugger> Run*. Убедитесь, что происходит передача управления на вектор прерывания. Зафиксируйте в отчете содержимое регистров TMR0, INTCON и стека. Выполните подпрограмму обработки прерывания в пошаговом режиме. Проконтролируйте, как происходит сохранение и восстановление контекста. По показаниям секундомера определите интервалы времени между последовательными прерываниями. Измените адрес точки останова так, чтобы можно было определить время выполнения цикла программы. Занесите результаты измерений в отчет.

2.3.6 Используя кабели, подключите:

- к разъему порта PORTC отладочного стенда периферийный модуль с установленными кнопками и часами реального времени;

- к разъему порта PORTD отладочного стенда периферийный модуль с установленными светоизлучающими диодами. Подключите отладочный стенд к USB порту персональному компьютеру.

Активируйте отладчик MPLAB IDE для работы с PICkit2 (команда *Debugger>Select Tool>PICkit2*). Если в окне *Output* на закладке *MPLAB PICkit2* появится сообщение об ошибке, проверьте разъемы и соединительные кабели.

После установления связи и самотестирования выполните загрузку целевого приложения в микроконтроллер отладочного стенда по команде *Debugger>Program*.

Нажмите кнопку панели *Run* или выберите команду *Debugger>Run*. По состоянию линейки светодиодов, подключенных к выводам PORTD, проверьте реакцию МК на нажатие кнопки. Сравните с заданным поведением. Нажмите кнопку панели *Halt* или выберите команду *Debugger>Halt*, чтобы остановить выполнение программы. Сбросьте МК, выбрав *Debugger> Reset> Processor Reset*.

2.3.7 Если результаты проверки Вас разочаровали, то в данном случае, вероятнее всего, причина в настройке портов или дребезге контактов. Проверьте правильность управляющих слов, загруженных в регистры TRISC, TRISD, OPTION_REG. Проверьте правильность реализации функции подавления дребезга контактов. После исправления ошибок и трансляции программы, повторите загрузку целевого приложения в МК и тестирование программы.

В случае успеха отключите отладчик системы MPLAB IDE командой *Debugger> Select Tool>None*. Отключите кабель от USB-порта компьютера. Закройте проект командой *Project>Close*. В противном случае необходимо локализовать и устранить ошибки в программе.

2.4 Оформление работы

Отчет по лабораторной работе должен содержать:

- схему алгоритма программы;
- исходный текст программы;
- расчетные и измеренные значения задержек;
- выводы.

2.5 Контрольные вопросы

Поясните по схеме алгоритма работу программы.

Как выполняется отладка программ, имеющих сложную структуру?

Как в среде симулятора можно имитировать изменения внешних сигналов на линиях портов ввода МК?

Каким образом выполняется программное подавление «дребезга контактов»?

Поясните, каким образом можно настроить таймер TMR0 на требуемый режим работы.

Как выполняется обработка запросов прерываний?

Как можно выполнять автономную отладку программ, использующих механизм прерываний?

Каким образом можно оценить длительность формируемых временных интервалов с использованием симулятора?

Перечислите основные режимы работы отладчика системы MPLAB IDE. Как выполняется настройка среды MPLAB IDE на требуемый режим отладки?

Каким образом можно отслеживать выполнение программы с использованием внутрисхемного отладчика PICkit2?

3 Отладка программы вывода символьной информации

3.1 Описание работы

Цель работы - получить практические навыки выполнения автономной и комплексной отладки программы вывода символьной информации.

Объектом отладки в работе является программа, которая должна выполнять модификацию содержимого ячейки Cnt. Содержимое ячейки Cnt должно отображаться на 7-сегментных индикаторах. Для управления индикаторами в отладочном стенде применен режим динамического управления.

Так как основной модуль программы несущественно отличается от уже отлаженного модуля программы в лабораторной работе №2, то целесообразно использовать технологию восходящей отладки. При этом сначала выполняется контроль работы подпрограмм B2_BCD, Pack, Dis7seg, а затем отладка основного модуля программы. Предполагается, что подпрограммы INT и D5ms отлажены при выполнении лабораторной работы №2.

3.2 Создание проекта и трансляция программы

3.2.1 В папке PIC создайте свою рабочую папку. Скопируйте в нее подготовленный файл с исходным текстом программы lr3.asm. Создайте новый проект с помощью мастера *Project>Project Wizard*. Задайте имя проекта lr3, включите в проект исходный текст программы lr3.asm.

Откройте окно с исходным текстом программы. Выполните трансляцию программы (команда *Project>Make*). Если программа содержала синтаксические и другие формальные ошибки, исправьте их и повторите трансляцию.

3.3 Отладка программы

3.3.1 Для тестирование подпрограмм символьного вывода вставьте перед инструкцией вызова функции B2_BCD фрагмент кода:

```
movlw   0xFF
movwf   L_byte
```

```
movlw   0x0F
movwf   H_byte
```

После отладки подпрограмм эти операторы необходимо будет удалить.

Для тестирования функций B2_BCD и Pack:

- создайте окно наблюдения и включите в него ячейки H_byte, L_byte, R0, R1, R2, Dig1, Dig2, Dig3, Dig4;

- установите точки останова на метках B2_BCD, Pack, Dis7seg;

- выполните прогон программы до точки останова B2_BCD, проконтролируйте значения переменных H_byte, L_byte;

- выполните прогон программы до точки останова Pack, проконтролируйте значения переменных R0, R1, R2;

- выполните прогон программы до точки останова Dis7seg, проконтролируйте значения переменных Dig1, Dig2, Dig3, Dig4.

3.3.2 Для тестирования подпрограммы Dis7seg откройте окно логического анализатора командой *View>Logic Analyzer*, нажмите кнопку *Channals* и в открывшемся окне *Configure Channals* нажмите кнопку *Configure Bus(s)*. В открывшемся окне *Configure Bus* нажмите кнопку *New Bus*, в окне *Bus Name* задайте имя группы (шины) – PORTB, выберите сигналы на линиях RB0 – RB3. Задайте имя группы (шины) – PORTD, выберите сигналы на линиях RD0 - RD7. подтвердите выбор нажатием кнопки OK. В окне *Configure Channals* выберите группы PORTB, PORTD, подтвердите выбор нажатием кнопки OK. Установить переключатель *Trigger Position* на *Start*.

3.3.3 Удалите точки останова с меток B2_BCD и Pack. В соответствии с планом тестирования для вывода четырех символов на индикаторы необходимо выполнить 4 полных цикла программы. Чтобы наблюдать в окне логического анализатора последний результат, необходимо выполнить дополнительный цикл.

Наиболее просто это обеспечить, используя сложные условия останова отладчика системы MPLAB IDE. Выполните команду *Debugger>Complex Breakpoints*. В открывшемся окне *Simulator Complex Breakpoints* нажмите кнопку *Add Breakpoint*. В

окне *Set Breakpoint* на закладке *Program Memory* необходимо в окне *Address* задать метку точки останова Bypass. В окне *Pass Count* выберите условие *Event must occur Count times*, в окне *Count* задайте число событий до останова программы.

Запустите выполнение программы в автоматическом режиме. После останова программы, используя окно логического анализатора, исследуйте временные диаграммы зарегистрированных сигналов. Убедитесь, что смена значений выходных сигналов на линиях PORTB и PORTD происходит в соответствии с планом тестирования. Скопируйте временные диаграммы в отчет. Определите период регенерации изображения на индикаторах. Удалите из текста программы отладочные инструкции. Выполните трансляцию программы.

3.3.4 Используя кабели, подключите:

- к разъему порта PORTC отладочного стенда периферийный модуль с установленными кнопками и часами реального времени;

- к разъемам портов PORTB и PORTD отладочного стенда периферийный модуль с установленными 7-сегментными индикаторами. Подключите отладочный стенд к USB порту персональному компьютеру.

Активируйте отладчик MPLAB IDE для работы с PICkit2 (команда *Debugger>Select Tool>PICkit2*). Если в окне *Output* на закладке *MPLAB PICkit2* появится сообщение об ошибке, проверьте разъемы и соединительные кабели.

После установления связи и самотестирования выполните загрузку целевого приложения в микроконтроллер отладочного стенда по команде *Debugger>Program*.

3.3.5 Активируйте выполнение программы в режиме реального времени. Проверьте реакцию МК на нажатие кнопки, фиксируйте результат на 7-сегментных индикаторах. Проконтролируйте правильность отображения информации. Нажмите кнопку панели *Halt* или выберите команду *Debugger>Halt*, чтобы остановить выполнение программы.

В случае успеха отключите отладчик системы MPLAB IDE командой *Debugger> Select Tool>None*. Отключите кабель от USB-порта компьютера. Закройте про-

ект командой *Project>Close*. В противном случае необходимо локализовать и устранить ошибки в программе.

3.4 Оформление работы

Отчет по лабораторной работе должен содержать:

- схему алгоритма программы;
- исходный текст программы;
- синтез управляющих слов;
- временные диаграммы сигналов;
- результаты тестирования программы;
- выводы.

3.5 Контрольные вопросы

Поясните по схеме алгоритма работу программы.

Каким образом выполняется вывод информации на 7-сегментные индикаторы отладочного стенда?

Как можно проконтролировать корректность работы подпрограммы вывода на 7-сегментные индикаторы?

Чем отличаются технологии восходящей и нисходящей отладки?

Как выполняется настройка логического анализатора системы MPLAB IDE?

Чем отличаются сложные точки останова от простых? В каких случаях их целесообразно использовать?

Перечислите использованные средства отладчика, обеспечивающие выполнение условий управляемости и наблюдаемости.

4 Отладка программы аналогового ввода

4.1 Описание работы

Объектом отладки в работе является программа, которая выполняет запуск модуля АЦП, обрабатывает результаты преобразования и выводит их на 7-сегментные индикаторы. Второй функцией программы является анализ состояния линии порта RC1. Нажатие с последующим отпусканием кнопки, подключенной к линии RC1, вызывает модификацию переменной Cnt. Значение трех младших разрядов Cnt выводится на линии RE2 – RE0 порта PORTE.

4.2 Создание проекта и трансляция программы

4.2.1 В папке PIC создайте свою рабочую папку. Скопируйте в нее подготовленный файл с исходным текстом программы lr4.asm. Создайте новый проект с помощью мастера *Project>Project Wizard*. Задайте имя проекта lr4, включите в проект исходный текст программы lr4.asm.

4.2.2 Откройте окно с исходным текстом программы. Выполните трансляцию программы (команда *Project>Make*). Если программа содержала синтаксические и другие формальные ошибки, исправьте их и повторите трансляцию.

4.3 Отладка программы

4.3.1 Проверку работы программы, поиск и исправление возможных ошибок выполняют в ходе отладки. Активируйте инструмент *MPLAB-SIM Simulator* командой *Debugger>Select Tool>MPLAB SIM*. Настройте симулятор для выполнения автономной отладки данного приложения (команда *Debugger> Setting*). В окне *Simulator Setting* на закладке *Osc/Trace* задайте значение тактовой частоты *Processor Frequency* 12 MHz.

4.3.2 Сформируйте удобную среду для отладки программы. Для автономной отладки программы необходимо иметь средства имитации аналого-цифрового преобразования. Для этих целей можно использовать стимулы регистра типа *Register*

Injection. Текстовый файл стимула регистра состоит из колонки значений, которые будут поочередно передаваться паре регистров ADRESH:ADRESL с учетом заданного выравнивания, когда аналого-цифровое преобразование будет выполнено.

Создайте новый текстовый файл - *File > New File*. В безымянном окне в одну колонку введите список из восьми 16-ричных значений результатов 10-разрядного аналого-цифрового преобразования, которые Вы хотите назначить регистрам ADRESH:ADRESL. Подберите эти значения, учитывая возможность контроля обработки результатов аналого-цифрового преобразования программой. Сохраните созданный файл - *File > Save As…* Выберите директорию, в которой хотите сохранить файл, введите имя файла с расширением txt.

Выберите пункт меню *Debugger>Stimulus>New Workbook* для открытия диалогового окна настройки стимула регистра. В открывшемся окне выберите закладку *Register Injection*. В указанных столбцах задайте значения:

- *Reg/Var* - ADRESL;
- *Trigger* – Demand;
- *Data Filename* – имя файла стимула;
- *Wrap* – Yes;
- *Format* - Hex.

Нажмите кнопку *Apply* , затем *Save* – сохраните стимул регистра.

4.3.3 Контролировать текущие значения сигналов можно в окне наблюдения. Выполните команду *View> Watch*, чтобы создать окно наблюдения интересующих ячеек памяти данных: ADRESH, ADRESL, H_byte, L_byte.

Чтобы получить информацию об изменениях сигналов на выводах МК можно использовать логический анализатор отладчика системы MPLAB IDE. Запуск логического анализатора выполняется по условию совпадения программного счетчика с заданным значением (адресом).

4.3.4 Настройте сложную точку останова для контроля работы программы в соответствии с планом тестирования. Настройте логический анализатор на отображение портов PORTB и PORTD.

4.3.5 Запустите выполнение программы в автоматическом режиме. После останова программы, используя окно логического анализатора, исследуйте временные диаграммы зарегистрированных сигналов. Убедитесь, что смена значений выходных сигналов на линиях PORTB и PORTD происходит в соответствии с планом тестирования.

При оценке правильности полученных результатов следует учитывать, что выравнивание результата аналого-цифрового преобразования и отбрасывание двух младших разрядов эквивалентно делению заданного в файле стимула значения на 4.

Установите точки останова на операторах выбора аналоговых каналов и операторах запуска преобразования. Используя секундомер *Debugger> Stopwatch*, оцените интервалы времени, необходимые для нормальной работы модуля АЦП, а также период дискретизации аналоговых сигналов. В случае необходимости скорректируйте длительности задержек.

4.3.6 Используя кабели, подключите:

- к разъему порта PORTA отладочного стенда потенциометр;

- к разъему порта PORTC отладочного стенда периферийный модуль с установленными кнопками и часами реального времени;

- к разъемам портов PORTB и PORTD отладочного стенда периферийный модуль с установленными 7-сегментными индикаторами. Подключите отладочный стенд к USB порту персональному компьютеру

- к разъему порта PORTE отладочного стенда периферийный модуль с установленными светоизлучающими диодами.

Активируйте отладчик MPLAB IDE для работы с PICkit2 (команда *Debugger>Select Tool>PICkit2*). Если в окне *Output* на закладке *MPLAB PICkit2* появится сообщение об ошибке, проверьте разъемы и соединительные кабели.

После установления связи и самотестирования выполните загрузку целевого приложения в микроконтроллер отладочного стенда по команде *Debugger>Program*.

4.3.7 Активируйте выполнение программы в режиме реального времени. Проверьте реакцию МК на нажатие кнопки, фиксируйте результат на светоизлучающих

диодах. Меняя положение оси потенциометра, проконтролируйте результат аналого-цифрового преобразования на 7-сегментных индикаторах. Нажмите кнопку панели *Halt* или выберите команду *Debugger>Halt*, чтобы остановить выполнение программы.

В случае успеха отключите отладчик системы MPLAB IDE командой *Debugger> Select Tool>None*. Отключите кабель от USB-порта компьютера. Закройте проект командой *Project>Close*. В противном случае необходимо локализовать и устранить ошибки в программе.

4.4 Оформление работы

Отчет по лабораторной работе должен содержать:

- схему алгоритма и исходный текст программы;

- результаты тестирования программы (временные диаграммы, значения временных интервалов);

- выводы.

4.5 Контрольные вопросы

Поясните по схеме алгоритма работу программы.

Как при автономной отладке можно проконтролировать корректность взаимодействия с модулем АЦП?

Как в среде симулятора можно имитировать выполнение аналого-цифрового преобразования?

Список использованных источников

1 PIC16F87X [Электронный ресурс]: Техническое описание. – М.: ООО «Микро-Чип», 2001. – 61 с. – Режим доступа: http://www.microchip.ru/files/d-sheets-rus/pic16f87x.pdf /. – 27.04.2018.

1 MPLAB IDE [Электронный ресурс]: User's Guide DS51519B.– Microchip, 2006. – 277 р. – Режим доступа: http://www.cis.upenn.edu/~lee/06cse480/data/MPLAB_IDE_User_guide.pdf/.– 27.04.2018.